High Energy Accelerator Simulation and Parallel Computing *

A.U. Luccio, N.L. D'Imperio
Brookhaven National Laboratory, Upton, NY
J.D. Galambos, OakRidge National Laboratory, OR, TN

August 16, 2000

Abstract

Simulation of the dynamics of a particle beam in an accelerator can be done by using a "herd" of representative macroparticles randomly generated. Particles are propagated through the accelerator lattice using transfer maps, consisting of matrices in the six dimensional particle phase space, and higher order transformations. In the presence of space charge effects, i.e. for high intensity beams, the interaction between macroparticles and between macros and accelerator chamber walls becomes important and in some case is the dominating effect, leading to the formation of beam halo. Computing can be very demanding and time consuming. Parallel computing is the only practical solution. We will present results and comparison for parallel computing using both an IBM cluster and a PC farm for two problems: simulation with the code *Orbit* of the intense proton beam in the Oak Ridge Spallation Neutron Source 1 GeV accumulator ring, and of the polarized proton beam for the RHIC collider at Brookhaven with the code *Spink*.

1 Accelerator simulation

Numerical simulation of particle accelerators using representative "macro particles" is important for: (i) the design of new accelerators, (ii) understand and optimize existing accelerators, and (iii) model-based control of accelerators. Accelerator simulation involve the repeated solution of equations of the dynamics of a "herd" of relativistic particles in electro-magnetic fields, produced by accelerator modules (magnets and RF cavities) and by the particles themselves (space charge forces in the presence of accelerator chamber walls).

*Work performed under the auspices of the U.S.Department of Energy.

1.1 Design of new accelerators

Similarly to what is being done in other fields of technology, like in the aircraft industry, the design of new accelerators make extensive use of computer simulation. In the US important examples of high current (space charge dominated) and high energy accelerators are

• The Spallation Neutron Source[1]. The SNS is a new 1.36 Billion facility funded by DOE, to be built at Oak Ridge National Laboratory. It will provide high-energy protons to be sent against a Mercury target to generate, in turn, 2 MW neutron bursts for research. The SNS is being designed and built as a collaboration between national laboratories. Its accelerator components are a 1 GeV proton linac, followed by a 1 GeV proton accumulator. Los Alamos is in charge of the linac, Brookhaven of the accumulator ring. A very high beam current characterizes the accumulator. One of the challenges is to minimize beam losses. Fig. 1 shows the SNS installation, its linac, ring, and target to produce neutrons.

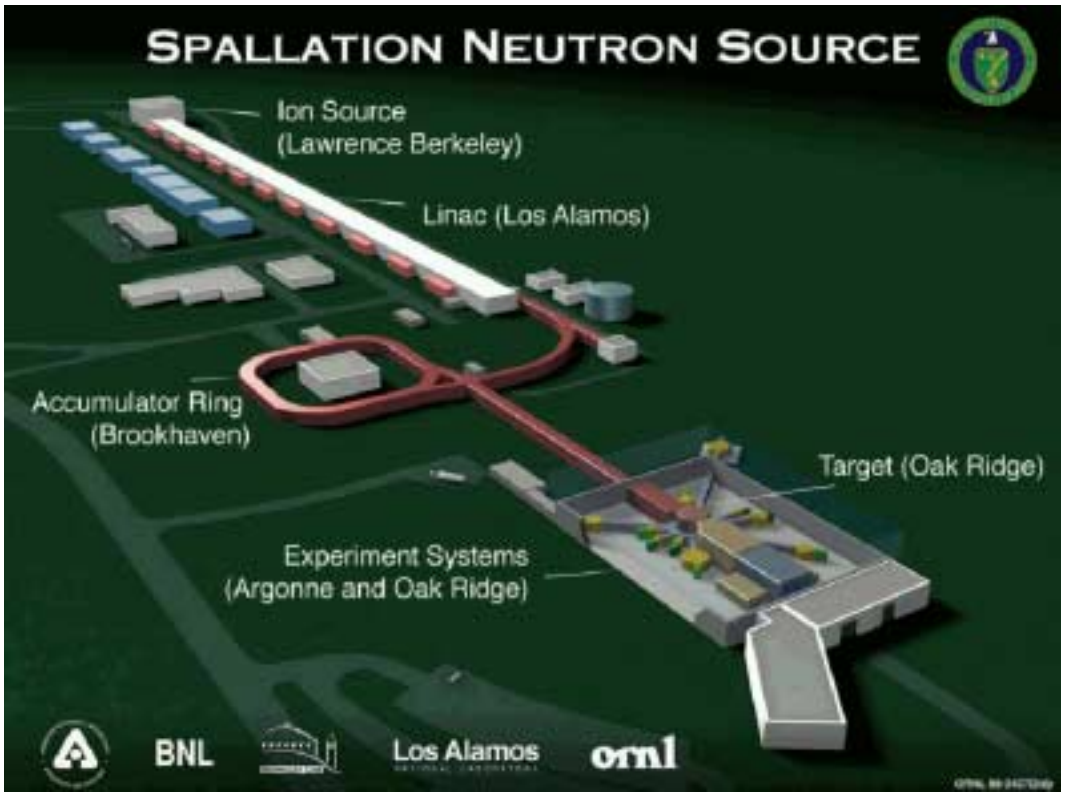

Figure 1: The SNS accelerator complex at Oak Ridge.

• The Relativistic Heavy Ion Collider, RHIC[2]. RHIC consistes of two superconducting counter-rotating rings, that intersects in six points. Particles (250

GeV protons and heavy ions up to Gold, 100 GeV/u) are brought to collisions in these intersections. Fig. 2 shows the RHIC accelerator complex, consisting of a cascade of various accelerators.

Figure 2: The RHIC accelerator complex at Brookhaven.

• The Muon-Muon Collider[3]. This is the ultimate machine for basic particle physics research being studied in various US laboratories, to collide high energy beams of muons against each other.

1.2 Optimization of the operation of accelerators

An example: the Alternating Ring Synchrotron at Brookhaven. The AGS is the highest intensity proton machine in the world. The machine also accelerates heavy ions and is presently used as the injector for RHIC. Although extensively and continuously studied for decades, the AGS still offers wide margins of improvements, to obtain higher currents under low beam losses. Beam simulation is an essential tool for this task. It helps understand the behavior of the beam, and suggests different operation modes.
To give some feeling of the problems encountered in simulate a synchrotron like the AGS for studies and optimization, look at Fig. 3. This figure represent real data for a Mountain Plot for the AGS Booster, i.e. a plot of the longitudinal measured profile of the beam for successive turns. The beam profile is controlled

by the shape of the RF wave, that in this case has a fundamental plus a second harmonic. A tracking program like *Orbit*, described later in this paper, can reproduce the mountain Range and explain some of its details

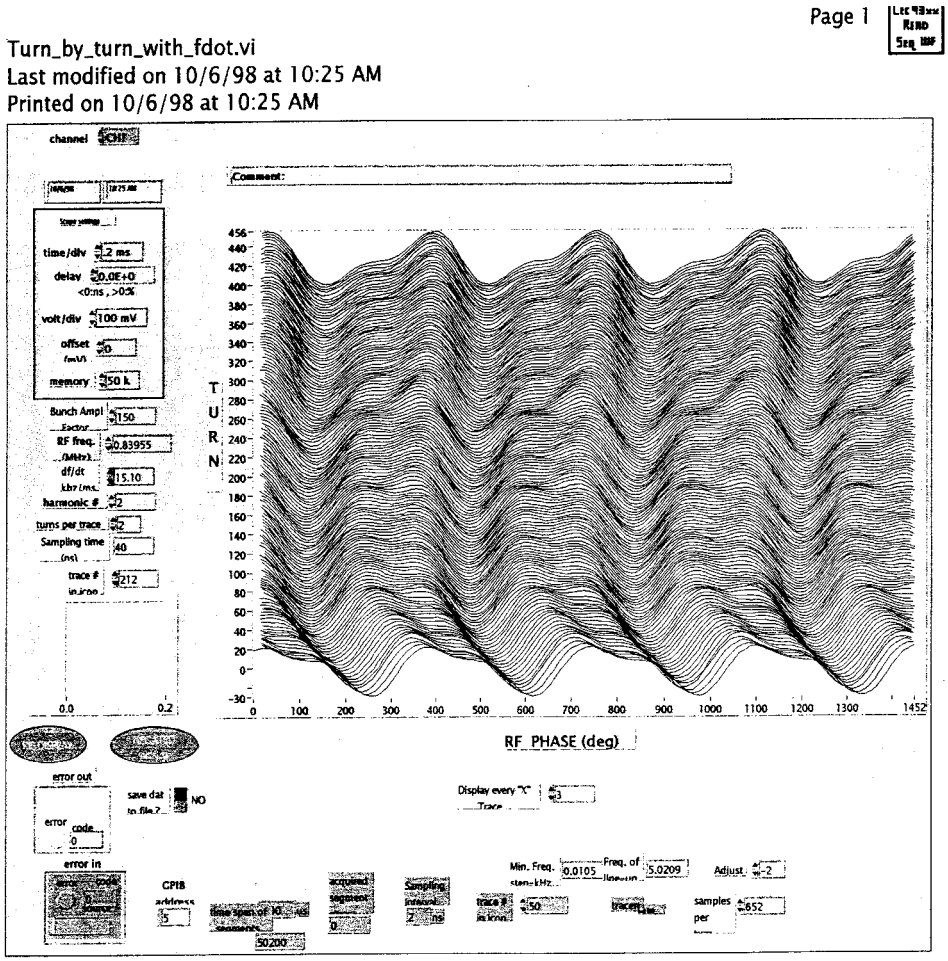

Figure 3: An experimental Mountain Range plot for the AGS Booster.

1.3 Model based accelerator control

Computers increasingly control accelerators. The control system is based on models, therefore on simulation. A computer environment is being created to integrate models and codes in a single unity. Extensive studies for accelerator control are underway for several machine in different laboratories.
At Brookhaven, for the AGS and RHIC the model based control has reached a high level of sophistication. Likewise, for the SNS accumulator ring, an im-

portant area of activity is the implementation of the model based control system.

An ideal model based control of any machine operation is one in which the operator should not in principle be aware whether he is working on the real or the model machine. Therefore the model should run in real time, which is difficult for an accelerator, that moves on very fast. For instance, in a synchrotron, the whole cycle of acceleration takes place in a fraction of a second, and in a cycle the beam performs thousands of turns. So, one is obliged to average data taken from the machine as input and commands issued to the machine as output of the control software over many turns. Even so, the challenges on the speed of running simulation codes are great.

2 Accelerator components

The physics of an accelerator is based on the study of the relativistic equations of motion of a particle in an electric and magnetic field

$$\begin{array}{c} \frac{d(\vec{\beta})}{dt} = \frac{q}{m\gamma c}\left[\vec{E} + c\vec{\beta}\times\vec{B} - \vec{\beta}\left(\vec{\beta}\cdot\vec{E}\right)\right] \\ \frac{d\gamma}{dt} = \frac{q}{mc}\vec{\beta}\cdot\vec{E} \end{array} \tag{1}$$

with the definitions

$$m = \gamma m_0,\ \vec{\beta} = \frac{\vec{v}}{c},\ \gamma = \frac{1}{\sqrt{1-\beta^2}} \tag{2}$$

and q and m the charge and mass of the particle, m_0 its rest mass, $mc^2 = \gamma m_0 c^2$ its total energy, $\vec{E}$ the electric field, $\vec{B}$ the magnetic field, and c the speed of light. From Eq. (2) note how the particle energy, proportional to γ, increases with its velocity. Equation (1) shows that to be effective to accelerate a particle on a curved trajectory, the electric field must have a non vanishing component parallel to the particle velocity and a magnetic field perpendicular to it.
Electro-magnetic fields are created by machine elements or modules, arranged in the lattice of the accelerator. The lattice contains magnets that provide magnetic fields $\vec{B}$, perpendicular to the orbit, and radio frequency cavities that provide electric fields $\vec{E}$, parallel to the orbit. Magnets can in turn be distinguished between (i) dipoles to bend the particle motion along a closed path, or to correct the orbit with angle kicks, and (ii) multi-pole magnet, that exert focusing forces on the particles.
Among multi-poles are the quadrupoles, equivalent to lenses for the particle beam, whose function is to maintain all the particles within the vacuum chamber of the accelerator, and sextupoles that provide for correction of particle of different energy within the beam (Chromaticity).
A general expression for the "strength" of a magnet of order n ($n = 0$ is a dipole) is

$$K_n = \frac{q}{p}\frac{\partial^n B_y}{\partial x^n} \tag{3}$$

where x and y are the transverse coordinate of the position of a particle along the accelerator with respect to a reference orbit. A multi-pole magnet imparts to the particle an angle kick proportional to K_n and to the n-th power of the displacement, x or y from the reference orbit. Quadrupoles ($n = 1$) give a linear kick and produce harmonic oscillations of the orbit.
Electric fields are provided by RF cavities to accelerate particle along their orbit. They increase the particle longitudinal momentum mv, by increasing their mass, Eq. (2), and their velocity (not to exceed the speed of light).

3 Accelerator models and algorithms

Accelerator models are based first on single particle dynamics, where particles don't interact with each other (low current machines or machines in the approximation of low current), and then on multi particle dynamics, dealing with particles interacting with each other and with the environment (accelerator chamber walls). Single particle calculations are needed to provide input to multi particle codes.

3.1 Single particle dynamics

A particle is represented by a vector in a 6 dimensional phase space

$$\vec{r} = (x, p_x, y, p_y, c\Delta t, \Delta p/p) \tag{4}$$

with x and y the transverse coordinates, p_x and p_y the components of the transverse momentum, Δt the deviation of the given particle, in time, from the "synchronous" or "ideal" particle, and Δp its deviation in longitudinal momentum (which is very close to the total momentum of the particle).
During the calculation, $\vec{r}$ is transformed from an accelerator module to the next, along the lattice of the machine by a map. This map must obey symplecticity conditions to insure that the relevant physical invariants are conserved. Important invariant is the Hamiltonian of the system, a function of the particle coordinates and of the electro-magnetic field through which they move

$$H = \sqrt{\frac{1}{c^2}\left(\mathcal{E} - q\phi\right)^2 - (mc)^2 - \left(\vec{p} - q\vec{A}\right)^2} \tag{5}$$

with ϕ the electric potential and A the magnetic potential. H represent the total energy of the system. To first order, a symplectic map is a matrix that represents a rotation of $\vec{r}$ in phase space.
The chain of transformations along the accelerator structure in the single particle approximation is calculated by *optics* codes. They take as input the physical parameters of the various accelerator modules, magnets, cavities and the like and their arrangement in the lattice and produce the transformation maps. They also decide if the lattice is stable and what the envelope of the beam is going to be in absence of inter particle forces. Among optics codes, well know

and extensively used are
• *Mad* (Methodical Accelerator Design)[4]. The CERN version can be downloaded from the Web. A different variation of *Mad* is also used at Brookhaven. The standard version is in Fortran77, another version in C++ is still experimental.
Mad allows one to design and optimize a machine lattice, by using a "thick" element description of accelerator modules. it is very popular to provide the lattice and the transformation maps to other codes. *Mad* contains tracking capabilities, but it is "static" in nature, i.e. it cannot describe some of the processes where lattice parameters are dynamically varied during tracking, e.g. the acceleration.
• *Marylie*[5]. From the University of Maryland, produces maps of high order for orbit transfer under symplectic conditions. These maps are calculated using Lie algebra methods.
• *Teapot*[6]. Developed at Cornell and extensively used at Brookhaven for the RHIC and SNS projects. It is a "thin" element code. *Teapot* forms the basis for the Accelerator Unified Library (AUL) database and control software[7].

An important feature of several optical codes, particularly highly developed in *Mad*, is "matching". It allows one to match the accelerator lattice to specific conditions, using numerical iterative algorithms of the Simplex, or Migrad type. Matching can be used to optimize the design of a lattice.

4 Multi particle dynamics. Tracking

A multitude of particles, or "herd", randomly generated is "injected" in the machine lattice generated by the optics code and their representative vector, defined in Eq.(4) is moved from a module to the next. Representative points are plotted at given intervals, or once per turn in a circular high energy accelerator. If the intensity of the beam is low, there are no collective effects and the particles in the beam propagate only in the external fields, independent of each other.

4.1 Tracking with Space charge

For high intensities, when space charge forces are present, tracking proceeds as follows: individual macro particle coordinates are transferred from a machine element to the next by maps, as individual particles. Particles are binned on a mesh to find the charge density $\rho(P)$. Space charge forces are then calculated and applied to each particles P producing an angle kick according to Eq. (8). And so on, through the next element..
A way to avoid repeating the binning process at each step is to transform the density ρ from element (lattice node) to the next, using the element maps. By treating the charge density like in an image in optics of light, it has been suggested to employ well developed numerical techniques involving, e.g. wavelets.
Another possible method is to scale, or "rubber-band" the density distribution according to the local envelope of the beam (as calculated in the single particle

approximation).
The properties and possible advantages of this methods in terms of computer speed have not been explored yet.

4.2 More on the space charge problem

Lets discuss to some extent the space charge problem, because it is very relevant to Parallel Computing.
Numerical handling of space charge is one of the most computer intensive problems in particle tracking[8]. It requires a simulation with many macro particles, each representing a multitude of real particles, the "herd", that continuously exert forces on each other due to their electric charge and current. Space charge is treated in the full 6-dimensional phase space of the particles (3 components of the position q and 3 of the momentum p) or only in the 4-dimensional transverse space with correction for the longitudinal dimensions.
The particle beam in a circular accelerator is generally a long structure that occupies a good fraction of the entire length of the accelerator (vacuum) chamber. Depending on the harmonic mode of the accelerating electric field there may be one or more beam "bunches" in a turn. The aspect ratio of a bunch is of mm to cm in the transverse x, y dimension to meters or tens of m in the longitudinal z. Individual particles perform oscillations within the beam around the equilibrium, or "design" orbit and along the length of the bunch.
Particles in the beam have the same electric charge, therefore they repel each other electrostatically. They are also equivalent to current elements, and these elementary currents attract each other. With considerations of elementary electrodynamics, by applying Gauss's and Ampere's laws to a small volume of space, the net force shows to be still repulsive, but decreasing as $1/\gamma^2$. This means that when the velocity of the particle approaches the speed of light c, the space charge force vanish, i.e. space charge forces are less and less important at high energies. Fig. 4 shows the force field in one transverse direction for a quasi-gaussian beam charge distribution, as calculated by *Orbit* (see Sect. 5).
Space charge treatment involves the inversion of the Poisson problem in the presence of boundaries. Intra-particle forces are calculated from the electric and magnetic fields generated by the beam on itself. Denoting the charge density with ρ, the potential ϕ in the beam, with the $1/\gamma^2$ factor, is calculated by solving the Poisson differential equation

$$\nabla^2 \Phi = -4\pi\rho \tag{6}$$

Poisson solvers can be written by using a "brute force" method, by direct integration over the space charge density

$$\phi(P) = \frac{C}{\gamma^2} \iiint \frac{\rho(Q)}{|P-Q| + \epsilon}\, dQ \tag{7}$$

where $Q = \vec{r}_s$ position of each particle, considered as a source (including image charges and currents induced on the walls), and $dQ = dx\, dy\, dz$ elementary

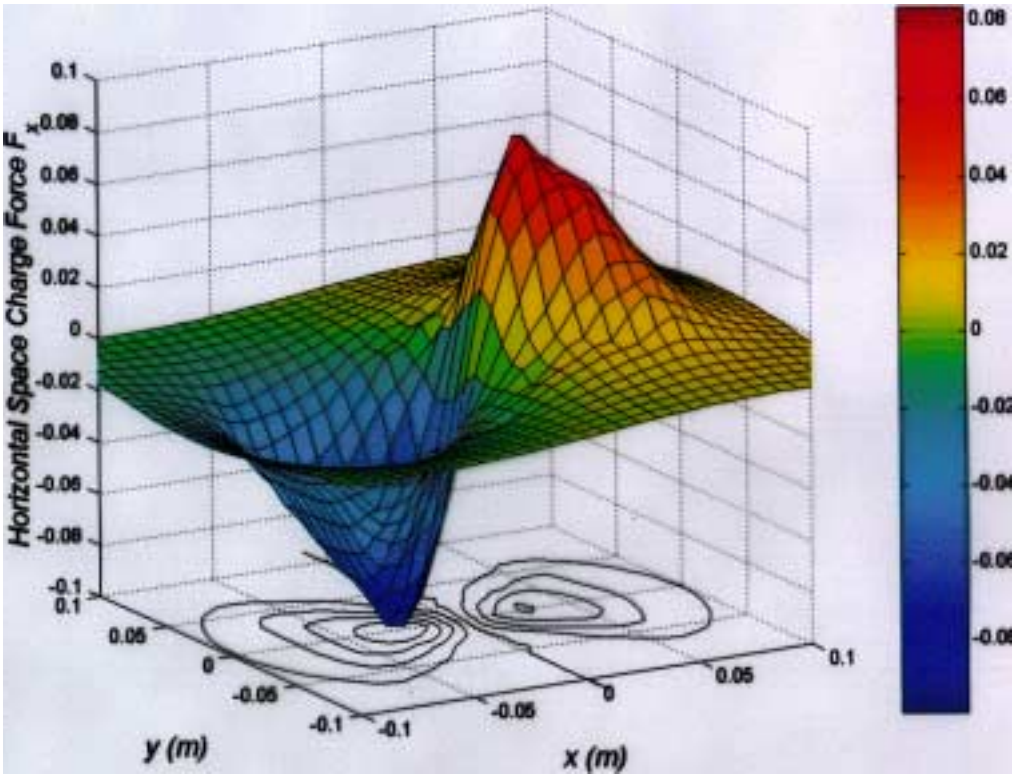

Figure 4: Horizontal component of the Space Charge force for a Gaussian beam.

source volume. Eq. (7) calculates the potential at each location $P = \vec{r}$, position of a particle acted upon by the field. ϵ is a smoothing factor to avoid infinities, and C a constant depending on the unities.
The Poisson equation can also be solved by transforming the integral of Eq. (7) to a convolution, by first taking a 2- or 3-dimension FFT of the charge density profile on a fixed or variable size mesh.
Once the potential, and then the forces by derivation, are calculated, a momentum kick is applied to each macro particle according to

$$\Delta\vec{p} = \int \vec{F}\, dt \tag{8}$$

Eq.(7) can be very lengthy to solve. An approximation is to integrate separately the transverse motion (x, y) and the longitudinal motion in z. I.e. to write the charge density as

$$\rho(x, y, z) = \rho_{\parallel}(z)\rho_{\perp}(x, y) \tag{9}$$

and treat $\rho_{\parallel}$ as a constant for a longitudinal slice of a beam bunch. The approximation is often acceptable, because in a circular accelerator the transverse motion of the particles within the beam (betatron oscillation) is much faster

than the longitudinal motion (synchrotron oscillation). With the position (9) the integral of Eq. (7) effectively loses a dimension. We may say that the code operates not in 3 but in "2-and-a-half" dimensions.

4.3 Poisson Solver based on LU Decomposition

Inversion of the Poisson equation in differential form can also be brought to the inversion of system of linear equations. Let us discretize the charge density ρ and the potential Φ at the corners of a mesh (i, j) and write Eq. (6) as

$$-4\pi\rho_{ij} = \mathcal{L}_{ij}^{kl}\Phi_{kl} \tag{10}$$

where $Q = (ij)$ is a discrete Source point and $P = (kl)$ a Field point, and the repetition of indeces implies a sum. The inversion of the Poisson equation is then performed as

$$\Phi(P) = -\frac{1}{4\pi}\mathcal{L}^{-1}\rho(Q) \tag{11}$$

with an inversion of the matrix $\mathcal{L}$. From the definition of the Laplacian (in 2 dimensions)

$$\nabla^2 \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{12}$$

$\mathcal{L}$ can be written using a difference expression for the second partial derivatives as

$$\mathcal{L}_{ij}^{kl} = -4\delta_i^k\delta_j^l + \delta_{i+1}^k\delta_j^l + \delta_{i-1}^k\delta_j^l + \delta_i^k\delta_{j+1}^l + \delta_i^k\delta_{j-1}^l \tag{13}$$

where $\delta = 0, \pm 1$ and the grid spacing is normalized to one. The Laplacian matrix $\mathcal{L}$ is large (size n^4), with n the number of grid points, but rather sparse. Because it is independent of the grid size, it has to be inverted only once, at the beginning of the tracking computation.
Several observations are in order.
• It is straightforward to express $\mathcal{L}$ in three dimensions, and treat with the LU method the space charge problem in 3D. The 3D problem can be thus naturally treated. In 3D, the matrix dimension will be n^6, rather formidable, however the matrix is sparse, as we noticed, but it ought to be inverted only once in vacuum. There are also several efficient algorithms to perform this inversion with parallel computation.
• If we are not happy with the linear difference expression of the partial derivatives in the Laplacian, we can use a higher order formulation on 7 or more grid points (10 or more in 3D). Of course, the matrix becomes a little less sparse.
• From the potential Φ the actual force can be immediately calculated by differentiation. In fact, we may choose to express directly the force field without calculating the potential first, as follows (in 2D)

$$\begin{aligned} F_x &= \tfrac{d}{x}\left(\mathcal{L}^{-1}\rho\right) = \left(\tfrac{d}{x}\mathcal{L}^{-1}\right)\rho \\ F_y &= \tfrac{d}{y}\left(\mathcal{L}^{-1}\rho\right) = \left(\tfrac{d}{y}\mathcal{L}^{-1}\right)\rho \end{aligned} \tag{14}$$

The differentiation is done with respect to the Field point coordinates and doesn't operate on ρ, that is a function of the Source point coordinates.
• When applying the above, a problem immediately arises. If the independent variable in tracking is time, then macroparticles in the herd that have started their journey together will arrive at a given time at different positions in the accelerator, because they may have different velocities or, mostly important in high energy machines, because they have followed paths of different length. If the independent variable is distance along the lattice, different particles will arrive at a given node at different time. The meaning of derivatives will change.

4.4 Walls and Impedances

The beam moves in an accelerator chamber where a high vacuum is maintained. Chamber walls are metallic or metal coated ceramic. The beam induces image charges and currents on the walls. They, in turn apply electric and magnetic forces on the beam. Moreover, fields remain for some time in the chamber after the beam is gone (wake fields) and in a circular accelerator can interact with the beam at successive turns, creating instabilities and beam break-up. Walls are reasonably smooth, but vacuum pump openings and other irregularities generate a very complicated environment where boundary become hard to code. A detailed study implies a careful modeling of the wall geometry, subdivided in many small partitions.
To reduce the burden of wall-to-beam interaction calculation, the effect of walls can be effectively studied by using effective impedances Z_n [9]. These are complex quantities, one for each n-th Fourier component of the beam density profile. Longitudinal and transverse impedance are calculated for a given wall geometry. They constitute the "impedance budget" for the accelerator. Wall induced additional (complex) voltage kicks on the beam are calculated in the frequency domain with

$$V_n = I_n Z_n \cos\left(n\omega t + \phi_n\right) \tag{15}$$

Kicks are thereafter transformed back to the time domain. The calculation of the impedance budget for a given geometry is *a priori* performed by specialized codes.

Longitudinal impedances are a more manageable set than transverse impedances. One reason is that in a conventional high energy accelerator the longitudinal motion of particles within the beam take place at a much lower frequency than in the transverse mode. Also, the beam is normally very long and interacts with a substantial fraction of the accelerator chamber length, averaging over many discontinuities. So, transverse motion requires a careful attention if we need to take into account the presence of walls.

We need to (i) invert the Poisson equation (6) with the appropriate boundary conditions at the walls, and (ii) treat the problem as much as 3D as we can, since walls act on each particle in the beam within a longitudinal length,

compatible with the requirement dictated by Relativity that the field emanated by the valls can reach the particle.
All Poisson inverters, based on direct integration (Brute Force), convolution (FFT), and Laplacian discrete inversion (LU) can deal with the problem. However, we have observed that LU seems more promising in dealing with 3D problems, then we elaborate on the latter. An important observation is that outside a contour that encompasses the totality of the beam charge, the field can be generally expressed as

$$\Phi = \Phi_0 \ln\left(\frac{r}{b}\right) + \sum_{n=1}^{\infty} \Phi_n \cos\left(n\theta + \psi_n\right) \left(\frac{b}{r}\right)^n + \sum_{n=1}^{\infty} V_n \cos\left(n\theta + \phi_n\right) \left(\frac{r}{b}\right)^n \quad (16)$$

Points on this contour can be added to the grid, and then the Laplacian will be inverted with the extra condition that the values on the contour should match the coefficients Φ_n and V_n needed for Eq. (16).
Using LU as described to deal with walls means that the Laplacian matrix is not the same for all nodes as in vacuum. It will be the same only for those space charge nodes that correspond to location in the accelerator where the vacuum vessel has the same shape. We will then have to invert a number of band-like sparse matrices at the beginning of computation and store all the inverted $\mathcal{L}^{-1}$. This can indeed be done off-line in a pre-processor parallel code, for all runs that use the same accelerator geometry.

4.5 Vlasov solver

The Vlasov or diffusion equation in principle contains everything one may want to know about the collective behavior of a multi particle system under the influence of electro-magnetic forces, either external or internal. It is a partial differential equation for the space charge density of the particles $\psi(q, p, t)$, with p and q the canonical position and momentum, and t the time

$$\frac{d\psi}{dt} = \frac{\partial\psi}{\partial t} + f\frac{\partial\psi}{\partial q} + g\frac{\partial\psi}{\partial p} = 0 \quad (17)$$

where f and g are the derivatives of the Hamiltonian, a function of the electromagnetic field

$$f = \frac{\partial H}{\partial p},\ g = -\frac{\partial H}{\partial q} \quad (18)$$

We don't know of any full dimensional Vlasov solvers in particle accelerator codes, only 2-dimensional.

5 Multi particle tracking codes

Several tracking codes have been written to follow a herd of interacting macro particles, generated at random over some prescribed distributions, through the lattice of the accelerator. These codes use maps generated by optics codes, like

Mad or *Marylie*, adding space-charge self-forces. Widely used, full 3-dimensional codes are

• *Orbit*[10], developed in a collaboration between Oak Ridge and Brookhaven National laboratories for the Spallation Neutron Source project. It is a C++, object oriented code. Fig. 5 shows a plot generated by *Orbit* of the beam of SNS in phase space, after a few hundred turns. Fig. 6 shows a second example of the beam longitudinal phase space foot print during acceleration (this was done when we were studying the possibility of ramping the energy of the SNS). The bucket contour is also shown.

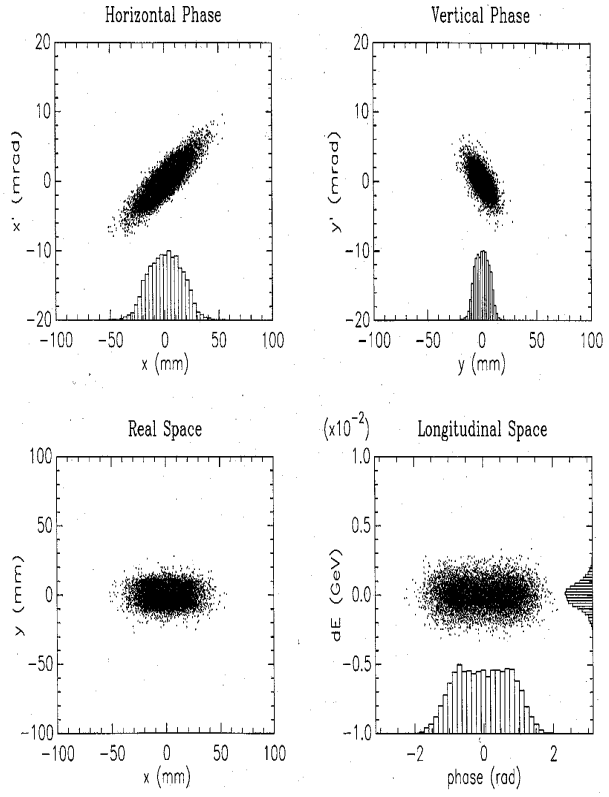

Figure 5: Phase space and real space footprints created by *Orbit*. No acceleration

• *Accsim*[11], in Fortran, from Triumf laboratory, BC, Canada. It was originally developed for the proposed Kaon Factory.

• *Simpsons*[12], in Fortran, from KEK, Japan. This code was originally developed for the defunct super collider in Texas, and is now being used for the

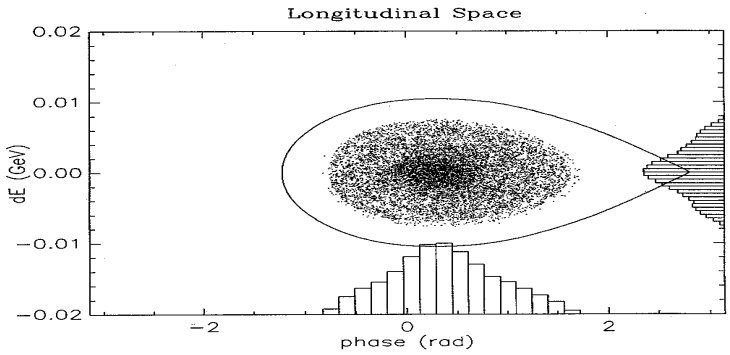

Figure 6: Longitudinal Phase space footprint with acceleration, created by *Orbit*.

Japanese Neutron Spallation Source project.
• *Warp*[13], in Fortran. Developed and used at the Lawrence Livermore Lab in California for the Heavy-ion Beam-driven Inertial confinement Fusion (HIF) project.
• *Track2D* and *Track3D*[14]. In Fortran, from the Rutherford Appleton laboratory, UK. Developed in conjunction with the European neutron spallation project.
• *Impact* [15], in C++, object oriented. Written in Los Alamos as part of a DOE Grand Challenge in Computational Accelerator Physics to simulate linear accelerators.
• *Spink*[16]. Developed at BNL to study the acceleration of polarized particles in the AGS and RHIC. *Spink* provides three more dimensions to the representation of a particle, i.e. the components of its spin.

An example of multi particle tracking with space charge is shown in Fig. 7, produced by *Accsim*. The five sections of the figure show, in the first row from left to right, the transverse phase space density (horizontal and vertical). In the second row the transverse "real" space (x, y) is shown and then the longitudinal space, and above the latter the distribution of the longitudinal space charge

force due to a conductive pipe surrounding the beam.

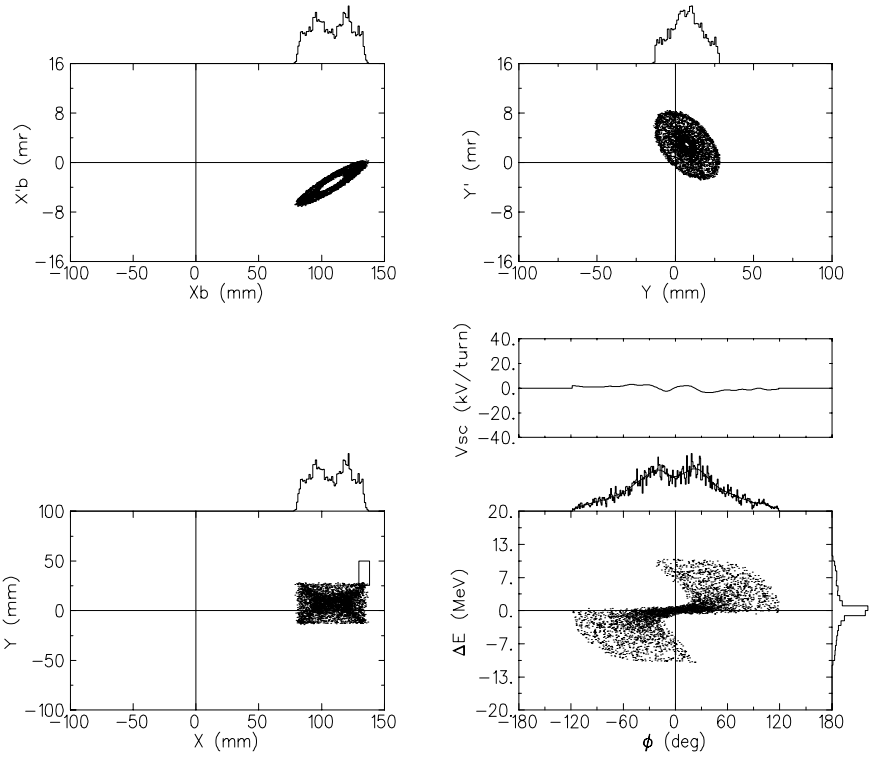

Figure 7: Example of tracking with *Accsim*

6 Other accelerator problems and issues

Particle accelerator modeling implies many issues to be addressed. Some may require intensive computing. Let us name a few

• Relativity: Special Relativity is always present in the formulation of the equations for the motion of particles in the high energy accelerators of interest here, since the velocity of the particles is close to the speed of light ($\beta \approx 1$). This has in particular relevance in the treatment of inter-particle forces, since the field propagate from particle to particle at the speed of light, which is comparable to the speed of the particles themselves (problem of the light cone).

• Independent variable: Space, time? Already mentioned. Most tracking codes used for high energy accelerators use the longitudinal position of the particle along its path, s, as the independent variable. This is convenient, since the description of the lattice is commonly made with a sequence of machine elements or modules ordered along s. However, for the solution of electro-magnetic problems, time is a more natural independent variable. Warp and Impact use time.

• Instabilities: The particles in an accelerator, under the effect of focusing forces perform oscillations around an equilibrium "reference" orbit. In a circular accelerator the frequency of these oscillations is called the betatron tune. Another tune, the longitudinal or synchrotron tune, is the frequency of oscillations in the longitudinal direction, in time and energy around a reference (synchronous) particle. Space charge forces affect the oscillations and modify the tunes.
If the tune is in resonance condition with some of the natural frequencies of the lattice, due e.g. with the periodic encounter by the particle of imperfections in the machine construction, the beam may become unstable and crash against the accelerator walls. Study of resonance instabilities by tracking is a very demanding task and computer intensive, requiring the numerical solution of both the transitory and asymptotic behavior of the integral of differential equations.

7 Computing

7.1 Methods, languages

In accelerator simulation the amount of computing can be overwhelming and often could only be one with parallel techniques. For tracking, we are always using different methods, alone or in combination
• Analytic. Accelerator theory is supported by experimental data that provides a backbone for computation. Models continuously use theoretical formulation to provide needed checks. Among others, important is that there are dynamical quantities, or "invariants", that must remain constant during tracking. We should watch them. Also, often there is a "core" of the beam that obeys a well known behavior and can be treated analytically, leaving to the numerics to deal with the details of the "halo".
Accordingly, it is tempting to write a code that at the same time does tracking and solves by symplectic integration some known equations, to control the behavior of the numerical results with the predictions of analysis.
• Particle-in-cell (PIC)[17]. It is one of the most used method to track. Macro particles are "injected' at random, with various statistical distributions, and thereafter tracked through the lattice. A reliable PIC model may require 10^5 to 10^7 macro particle to represent with good detail a typical halo. Also, if one wants a good representation of a complicated wall environment, one has to add a comparable number of wall partitions
• Contours. One can reduce the number of macro particles, putting them on contours of multi dimensional phase space volumes. Contours propagate in a well known way. This method reduces by one the dimensionality of the herd, however it requires a fine geometrical analysis, to insure among other things that particles are roughly equidistant (in, say, 5 dimensions), to avoid clustering that will decrease the accuracy of results. One ends up to the topological problem of the *Convex Simplex*.

High level language used are
• Fortran 77 or 90. Fortran is still very much used in the community of physicists, much because of its simplicity and because of the vast mathematical libraries available, but also because object oriented features may not be so important for accelerator tracking. Programs in Fortran are simple to develop and to debug.
• C++. When accelerator simulation is being used for accelerator control, the object-oriented features of C++ become important. C++ enables a natural interaction of the software with accelerator database and actual hardware. In C++ (as used in *Orbit*) classes describe accelerator elements and functions.
• *SuperCode*[18]. Due to the speed of modern interpreters, sometimes it proves convenient to have only part of the simulation code in a compiled form, and interpret part of the program. The driving script can be a C++ fragment that is being interpreted on the fly and can be modified during execution. This feature is very useful during test runs, to test rapidly different tracking scenarios. *Orbit* is currently compiled with *SuperCode* and is being run using a SC driver shell.

7.2 Parallel computing

Parallel computing is fast becoming a necessity for accelerator modeling, especially for quasi-real time accelerator control, where speed of response is a must, but also more in general to deal with PIC codes involving a large number of macro particles and a complex environment.
Parallelization is useful also in single particle optics code in optimization loops. After the basic parameters of an accelerator have been defined, the next exercise is to refine it to optimize some processes, like the injection or extraction strategy or determine the best position of orbit correction elements. The simulation code becomes then a subroutine to an optimization code that tries to vary parameters and find solutions by conjugate-gradients or other mathematical tools. It is obviously a very time consuming process that can be sped up in parallel.

At this point, it is important to make a distinction between linear accelerators and circular accelerators. Linacs are "single pass" devices. In them, a beam is injected, accelerated and extracted and doesn't pass twice through the same location in the accelerator lattice. Destructive instabilities, if any, develop very fast and with the characteristics of beam "break-up". In linacs, it is in general not too expensive to model in detail the configuration of the wall boundaries and use a very large herds of macro particles.
Comparatively, circular accelerators must track particles for many turns, ranging from about 10^3 in the SNS to a few 10^6 in storage rings like RHIC. Suppose that the lattice contains from 300 (SNS) to 3,000 (RHIC) maps and that the nodes at which space charge forces are updated are from 100 to 1,000 per turn, one can see how the amount of computing is large and one could barely afford the luxury of fine partitioning the walls or using big herds. In circular machines the instabilities may develop slowly and this *per se* require careful tracking in unstable regimes.

Some of the tracking problems for accelerator simulation are "embarrassing parallel". E.g. in a PIC code it is almost immediate to break up the herd of macro particles that constitute the beam in smaller herds, assigning every sub-herd to a "child" processor, with a "parent" processor polling results at given intervals or locations along the lattice to perform collective calculations, like for space charge. The efficiency of this procedure depends on the relative speed of performing given mathematical operations as compared with the time of exchanging informations among the nodes of the parallel computer. The net result is that in general, the total computer time is inversely proportional to the number of nodes only for a few nodes, and tends to saturate with increasing node number[19].
Less embarrassing and much more complex parallel computing issues for accelerator simulation involve the optimization of programming loops and procedures to solve specific problems, like Poisson or Vlasov solvers. The parallel optimization of the algorithms, in these cases, is based on the observation that solving Eqs. (6) or (17) can be reduced to the inversion of a linear operator for the specific accelerator node[20], that repeats from turn to turn, and ultimately to parallel matrix multiplication.

Parallel paradigms depend somewhat on the platform being used, on the operating system, on the popularity and support of a given language, and it is also a matter of personal taste. The most popular libraries are
• MPI (Message Passing Interface)[21],
• PVM (Parallel Virtual Machine)[22].

7.3 Graphics

Graphics representation of the results of simulation is very important. We can recognize three phases of the visualization process
• Post Mortem. Some of computer modeling graphics from simulation is performed after the calculations are done. At this point, data files are read and plotted. There will be always a need for this kind of post-mortem plotting.
• Interactive. It is highly desirable to have an interactive graphics that would show the results of a simulation, while the computing is in progress. In particular, using *SuperCode*, interaccting plotting would facilitate one to stop and restart calculation and change parameters to steer the results in the desired direction. A continuous pipe of calculation output to graphic is needed, that would also create an effective animation.
• Graphic firmware used for calculation. On some platforms, the graphic firmware is very efficient and fast. Some mathematical manipulations and even calculations can be effectively done using these features. Examples are the calculation of areas by pixel count and the mapping of particle densities with colors, where numerical codes associated with colors can directly enter into calculations.

7.4 Hardware

For sequential running of programs, workstations are the most used platforms. For parallel computing, PC clusters are rapidly becoming a popular choice, because of their limited cost and modularity. For accelerator simulation the best option at this time seems to have a number of nodes between 8 and 256, with a large distributed memory.
we should also mention the possibility of greatly speeding up computing by complementing a parallel structure with special hardware, that can solve directly a given problem like the MD-grape card[23]. This card calculates directly the forces acting among n-bodies, either molecules, planets or charged particles, and can be invoked by a code in place of a subroutine.

8 Examples of Implementation: *Orbit*

John Galambos implemented a parallelization with PVM of the tracking code *Orbit*, using up to 7 533MHz alpha chip processors[19]. In this simulation 100,000 macroparticles were used in a Spallation Neutron Source accumulator ring lattice. Transverse cpace charge was calculated with FFT on a 128x128 mesh.
We repeated his example on the Brookhaven Linux Cluster (BLC) with PVM and 4 processors (Pentium Pro), then we started to implement a MPI version of *Orbit*, first on an IBM SPARC machine with 32 processors and then on the CLC Cluster (PC Farm) at Brookhaven, using up to 36 children processors. The CLC is our cluster of choice. We are in the process to add 86 more processors to that cluster. During our timing tests we didn't share the cluster with any other user. With the CLC we used the option *-nolocal* with the *mpirun* command to restrict the operation of the parent (a much slower machine, as it can be seen from the table) to synchronization purposes. Running *Orbit* also on the parent resulted in considerably slowing down the computation.
Up to the maximum number of 32 processors, the computation speed (wall clock) of the CLC is proportional to the number of processors. Also the IBM SPARC speed is proportinal to te number of processors in dealing with the trivial problem in *Orbit*. The overall speed of this latter much more expensive machine is greater, due to the different architecture of the connections. Architecture and Communication Specifications of the IBM SPARC and the CLC Cluster are given in Table 8 Timing results of *Orbit* runs performed on the CLC cluster are given in Table 8. The total number of macros in the first 12 runs, with and without space charge calculation was $1.6\,10^6$. A further run with 25 million macros took somthing over 1,000 seconds. Running with continuous injection up to such many macros as a final number for 1,254 turns and using 26 processors would have ideally taken 3 days in the CLC cluster. A number of macros exceeding $25\,10^6$ is not at the present time possible due to memory limits.

Table 1: Comparison of Specifications for two Parallel Machines.

IBM v60 Cluster		
32 Nodes. Per Node:		
4 CPU's	4 Gigabytes Memory per Node	Shared Memory
CPU specs:	IBM Power3 Processors	375 MHz
1.5 GFlops/CPU - peak	64 kBytes L1 Cache	8 Mbytes L2 Cache
Interconnection:	SP Switch	TB3MX adapters
133 MB/s Bandwith per link	24 μsec latency	
File System:	GPFS Shared File System	
MPI:	IBM implementation of MPI	1.2 compliant
Brookhaven CLC Cluster		
Nodes (clc000):		
2 CPU's	256 MBytes RAM	
CPU specs:	Intel Pentium II (Klamath)	
300 MHz	512 kBytes L2 Cache	
Nodes (clc001-clc018):	2 CPU's	512 MBytes RAM
CPU specs:	Intel Pentium III (Katmai)	
500 MHz	512 kBytes L2 Cache	
Interconnection:	Cisco Catalyst 2900 Switch	100 MBit/sec Fast Ethernet
File System:	NFS	
MPI:	mpich implementation	1.2 compliant

Table 2: *Orbit* timing (wall clock) on the CLC Cluster. 1 Turn.

		With Space Charge	No Space Charge
Processors (parent + children)	nMacrosPerTurn per Node	Elapsed time sec	Elapsed time sec
2	$1.6\ 10^5$	189	61
3	$0.8\ 10^5$	105	30
5	$0.4\ 10^5$	68	17
9	$0.2\ 10^5$	48	9
17	$0.1\ 10^5$	43	6
33	$0.05\ 10^5$	37	3
2	$1.6\ 10^6$	1934	818
3	$0.8\ 10^6$	936	353
5	$0.4\ 10^6$	491	176
9	$0.2\ 10^6$	253	85
17	$0.1\ 10^6$	142	42
33	$0.05\ 10^6$	88	23
26	$25\ 10^6$	1161	

Comparable results obtained with the IBM machine are reported in Table 8. Clearly, the IBM machine is faster than the CLC, because of a much faster connection within the boxes and the partially shared memory, however, its speed dopesn't appear so proportional to the number of processors.

9 Another Example: *Spink*

Spink is a tracking code specialized in high energy polarized protons[16]. A proton in *Spink* has the 6 phase space coordinates shown in Eq. (4), as in *Orbit*-like tracking codes, plus 3 spin coordinates (the components of a real unitary vector, the spin vector). *Spink* reads the lattice and the transfer maps of a high energy accelerators and produces in addition spin matrices, that describe the rotation of the spin vector during particle circulation and acceleration that eventually may lead to loss of polarization. The rotation of the spin by an angle $\phi$is described with a 3x3 matrix

$$R = I\cos\phi + W\frac{1-\cos\phi}{\omega^2} + A\frac{\sin\phi}{\omega} \tag{19}$$

with I a unitary matrix, W a symmetric matrix, and A an antisymmetric matrix expressed as a function of the components of the precession axis. In

Table 3: *Orbit* timing (wall clock) on the IBM Cluster.

One Turn		With Space Charge	Space Charge Disabled
Nodes (parent + children)	nMacrosPerTurn per Child	Elapsed time sec	Elapsed time sec
2	$1.50\ 10^5$	48	18
3	$0.75\ 10^5$	24	8
4	$0.50\ 10^5$	16	5
6	$0.30\ 10^5$	13	3
16	$0.10\ 10^5$	16	1
2	$1.50\ 10^6$	564	251
3	$0.75\ 10^6$	281	118
4	$0.50\ 10^6$	209	91
6	$0.30\ 10^6$	124	51
16	$0.10\ 10^6$	42	12
2 Turns		With Space Charge	Space Charge Disabled
Nodes (parent + children)	nMacrosPerTurn per Child	Elapsed time sec	Elapsed time sec
2	$1.50\ 10^5$	143	48
3	$0.75\ 10^5$	70	22
4	$0.50\ 10^5$	47	14
6	$0.30\ 10^5$	31	8
16	$0.10\ 10^5$	14	7
2	$1.50\ 10^6$	1653	702
3	$0.75\ 10^6$	821	322
4	$0.50\ 10^6$	605	251
6	$0.30\ 10^6$	358	143
16	$0.10\ 10^6$	121	39

SPINK, while the orbit maps are considered "thick", the spin maps are considered "thin", producing an instantaneous spin precession.

In a collider like RHIC, that is the machine on which we are interested, the polarized proton beam has a relatively low intensity, so collective space charge effects are important only to describe beam-beam collisions between the two counter-rotating beams. However, since to well represent the beam one has to track many particles and for a very large number of turns, up to millions, parallel computing is important. Except for beam-beam effects, as noted, tracking of protons with spin is at first an embarassing problem, with each computer node tracking a subset of representative macroparticles, and the parent collecting and averaging data at given intervals.
Fig. 8 shows an example of spin tracking in RHIC. Crossing spin resonances produces some loss of polarization. The polarization is recovered though, because there are two Siberian Snakes in the ring. The figure shows two curves, calculated with a perfect lattice in RHIC, and with a lattice where the various machine element have some errors in position and field.
Fine and sometimes crucial details of the dynamics of the beam lead to algorithms that can be effectively dealt with in parallel. Some of these effects, that can be very important for the preservation of polarization at high energy arise from the fact that the lattice of the machine, in this case RHIC, can be varied while the beam is being tracked. The two main effects, non linear in nature, are related to the processes of (i) acceleration, and (ii) tune change.
To understand this point, recall that all transformation maps for the orbit come from optics programs, like *Mad* or *Teapot*, described earlier, that are static in nature. All machine functions that are building blocks of the algorithms being used in tracking of both orbits and spin are normally calculated in these codes and stored in some external file. If something is varied in the machine while the tracking takes place, as it automatically happens during the acceleration process or because we want to vary some parameters (e.g. the betatron tunes) on the fly, still the precalculated machine functions would not change, with results that may be then somewhat wrong. A strategy to correct for this behaviour is to dedicate a number of processors to dynamically change and update the machine lattice and pipe the results to *Spink* while other processors are doing the tracking.

10 Acknowledgments

We are indebted to Profs. James Glimm and Yuefan Deng of Stony Brook for stimulating discussions in particular on the LU problem. James Davenport of Brookhaven is helping us in pursuing parallel computing with the clusters of the Center for Data Intensive Computing. Still at Brookhaven, Mike Blaskiewicz, author of the FFT implementation of the Poisson solver in *Orbit*, Dan Abell, Alexei Fedotov, Nicolay Malitsky, smart users of *Orbit*, always participate in discussions on the code and on how to improve the physics in it. Robert Walkup

of IBM was invaluable for transforming *Orbit*, already made parallel with PVM by John Galambos into an IBM MPI code. At IBM, Gyan Bhanot showed us how to formulate in practice the LU Poisson Inversor. Jeff Holmes at Oak Ridge maintains *Orbit* and we use often his insight on how to improve and understand the algorithms.

References

[1] J.R.Alonso, for the SNS Collaboration *The Spallation Neutron Source Project* Proc. PAC99: 1999 Particle Accelerator Conference, Editors: A.Luccio and W.MacKay, IEEE 1999, p.574

[2] M. Harrison *The Commissioning Status of RHIC* Proc. PAC99, loc.cit., p.6, 1999

[3] K.T.McDonald *Muon Collides: Status of R & D and Future Plans* Proc. PAC99, loc.cit., p.310, 1999

[4] H.Grote and F.Ch Iselin *The Mad Program, Version 8.19* CERN/SL/90-13, Geneva 1996

[5] A.J. Dragt et al *Marylee 3.0 User's Manual* University of Maryland Physics Department Report 1999

[6] L.Schachinger and R.Talman *Manual for the Program Teapot. Noninteractive Fortran Version/* Particle Accelerators, 22, 35, 1987

[7] N.Malitsky and R.Talman AIP Conf. Proceedings 391, 1996

[8] A.U.Luccio and W-T Weng, Editors *Workshop on Space Charge Physics in High Intensity Hadron Rings* AIP Conf. Proc. 448, 1998

[9] A.W. Chao *Physics of Collective Beam Instabilities in High Energy Accelerators* Wiley, NY 1993

[10] J.D.Galambos, J.A.Holmes, D.K.Olsen, A.Luccio, J.Beebe-Wang *Orbit User Manual Version 1.10* SNS/ORNL/AP Technical Note 011, Rev.1, 1999

[11] F.W. Jones, G.H. Mackenzie and H. Schönauer *Accsim - A Program to Simulate the Accumulation of Intense Proton Beams* Particle Accelerators 31, p.199, 1990

[12] S.Machida et al. *Space Charge Effects in Low Energy Proton Synchrotrons* Nucl. Instrum Methods, A309 (1991) 43 also: *The Simpsons User's Manual* SuperCollider Laboratory, Dallas, TX, 1992

[13] A.Friedman, D.P.Grote and I.Haber *Three-dimensional particle simulation of heavy-ion fusion beams* Phys. Fluids B: Plasma Physics 4, 2203, 1992

[14] C.R. Prior *The Multi-Particle Code Track2D: A Guide for Users* CLRC, Rutherford Appleton Laboratory, 1993

[15] R.Ryne et al. *Program Impact* Proc. Linac Conf., Chicago, IL, 1998

[16] Alfredo U. Luccio Spin Tracking in RHIC (Code Spink) in: *Trends in Collider Spin Physics* World Scientific 1997, p.235 also: A.Luccio, A.Lehrach, J.Niederer, T.Roser, M.Syphers, and N.Tsoupas *New Capabilities of the Spin Tracking Code Spink* Proc. PAC99, loc.cit., p.1578, 1999

[17] R.W.Hockney and J.W.Eastwood *Computer Simulation Using Particles* Adam Hilger, IOP Publishing, New York, 1988

[18] S.W. Haney *Using and Programming the SuperCode* Lawrence Livermore Lab. Rept. July 21, 1995

[19] J.Galambos *Parallel Computing with Orbit* ORNL Accelerator Physics Memo, August 1999

[20] W.H.Press et al. *Numerical Recipes* Cambridge University Press 1994

[21] W.Gropp et al. *MPI: The Complete Reference* MIT Press, 1998

[22] A.Geist et al. *PVM: Parallel Virtual Machine* MIT Press, 1994

[23] T.Marumi, R.Susukita, T.Ebisuzaki, G.McNiven, B.Elmegreen *Molecular Dynamics Machine: Special-Purpose Computer for Molecular Dynamics Simulation* Molecular Simulation, 1999, Vol.21, pp.401-415

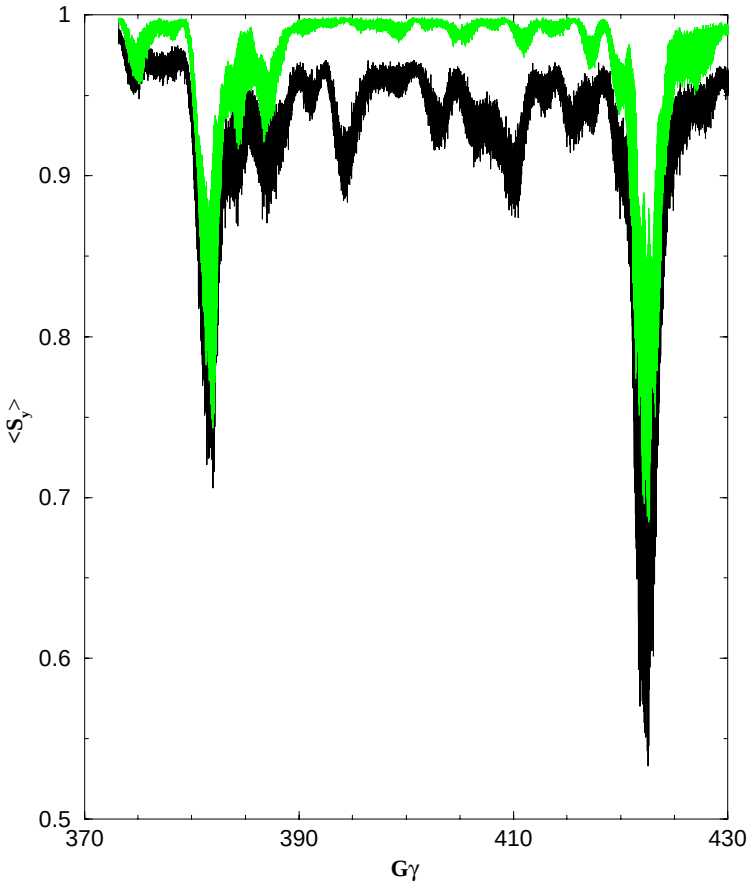

Figure 8: Example of spin tracking with *Spink*. The vertical component of the spin vector is shown in a RHIC lattice with and w/o errors.